明 細 書

並列処理システムの生成方法

技術分野

[0001] 本発明は、並列処理システムの設計に関し、定義ファイルに基づき並列処理システムのハードウェア構成情報を生成する方法および装置、さらに、定義ファイルに基づき並列処理システムをシミュレートする方法に関するものである。

背景技術

[0002] LSIあるいはASICを設計するために幾つかの言語が用いられている。C言語などの抽象度の高い高級言語と称されるものは、手続き的レベルの言語であり、1命令毎にいかに処理全体が順番に実行されていくかを示すのに適している。このレベルの記述は、一般的にハードウェア依存性がなく、適当なコンピュータで受け入れられるアプリケーションプログラムであり、LSIの仕様、またはLSIにおいて実行する処理全体を一般的に記述するために用いられる。Verilog−HDLあるいはVHDLなどのハードウェア記述言語(HDL)は、RTLと称されることもありレジスタトランスファレベルで、特定のハードウェアにより特定の命令が実行されているデータパスやパスを駆動するシーケンスを記述するために用いられる。

[0003] アルゴリズムは、問題を解くために明確に規定され、順序付けられた有限個の規則からなる集合として定義されており、従来、並列処理は、アルゴリズムにより記述された処理全体(アプリケーション)を順番に進める上で、独立して実行可能な部分(処理)を並列に実行し、処理時間を短縮する目的で用いられる。予め並列処理に適したハードウェアリソースを備えたシステムでアプリケーションを実行する場合は、コンパイラなどにより並列処理ができる部分は並列化され、実行速度を向上することがトライされる。

[0004] また、特定のアプリケーションの実行を目的としたハードウェアを設計する場合は、独立して実行できる部分を並列に処理するように回路を設計して処理時間の短縮を目指す。特開平10−116302号公報に記載された技術は、並列処理および同期通信等により実行時間が定まらない処理を記述可能なHDLにより回路を設計する方法

である。同期通信とは、2つのファンクションを並行に実行する際に、それらに含まれるプロセスを送信側が準備できるまで受信側が待ち、通信が完了した後にプロセスが進行する。したがって、これらのファンクションは並列に記述されても独立して実行されず、実行時間が可変の処理となる。一方、同期通信を行わない処理は並列処理として独立して行われる。これらは、ソース言語で与えられた処理全体の内、ソース上では並列に実行するように記載された処理を、ハードウェア設計において並列に、あるいは同期通信を用いて実行して実行サイクル数を短縮することを目的とした技術である。

[0005] 近年、LSIを構成する回路の一部をソフトウェアにより再構成できるハードウェアが提供されている。さらに、国際公開WO03／007155号には、再構成する基本単位を、ゲートレベルからALUなどのある程度の規模の演算機能を備えた演算ユニットにして、複数種類の演算ユニットをマトリクス状に配置し、再構成に要する時間を短縮することが開示されている。複数の演算ユニットがマトリクス状に配置されたシステムは、それぞれの演算ユニットが並列に処理を実行できるので、膨大な数の並列処理に適したハードウェアリソースを備えたシステムと捉えることが可能である。しかしながら、この種の並列処理に適したシステムを設計するのに適した設計システムは提供されていない。

[0006] C言語などのソフトウェア設計に適した高級言語は、アルゴリズムを、それに含まれる規則を時間的な順番に処理することを前提としている。したがって、プログラムカウンタを進めて命令がシーケンシャルに実行されるように構成されており、シーケンシャルではない並列という概念を導入することは難しい。命令を並列的に記述することが許容されたとしても、それは、時間的な順番に齟齬をきたさない範囲で、独立して実行できる処理を空間的に並列に広げて実行できる程度であり、並列処理に適したハードウェアリソースを積極的に使用することはできない。さらに、高級言語であれば、ハードウェアに依存しない命令が記述されるために、並列に記述した命令がハードウェアで実際に開始されたり、終了したりするタイミングは不明である。したがって、処理できる範囲を空間的に広げても、設計者はハードウェア上で実際にどのように並列処理が実行されるのかは定義できないし、把握することもできない。

[0007] HDLは独立して動作する回路構成を記述するので、本来、並列処理を記述するものである。また、ハードウェアが明確になるので、処理が実行されるタイミングを調べたり、調整できる。このため、高級言語で与えられたアルゴリズムを実現するHDLを記述することができる。しかしながら、逆に、特定のハードウェアを前提として記述されるので、汎用性はなく、ハードウェアが異なれば同一のアルゴリズムを実現することは不可能である。また、HDLが対象としているハードウェアが分からなければHDLに含まれているアルゴリズムを理解することもできない。

[0008] 高級言語を特定のハードウェア用にコンパイルした結果として得られるマイクロプログラムのレベルでは、完全に独立して実行可能な命令を並列に記述するVLIW技術、複数の命令を同時にフェッチして並列実行できる命令を見つけ出して実行するスーパスケーラ技術がある。これらは時間的に並んだ処理のうち、それを実行するために用意された複数のパイプラインで空間的に並列に実行できるものを並列に処理して実行速度を改善する技術であり、時間的な順番に齟齬がおきない範囲を空間的に広げる点では高級言語と変わりはない。すなわち、マイクロプログラムもプログラム言語であり、プログラムカウンタを進めて命令を順番に実行するシーケンシャルな処理はVLIWやスーパスケーラにおいても同様に必要とされる。さらに、マイクロプログラムは、特定のハードウェアを前提としており、HDLと同様に汎用性は少ない。

[0009] このように、プログラムカウンタを進めて実行するプログラム言語では、高級言語であれば汎用性があり、ソフトウェア設計は行い易いが、並列処理に展開できる部分はプログラムカウンタを進めても独立して実行できる処理に限られてしまい、多数の並列処理に適した演算ユニットを有効に利用することは難しい。ハードウェア依存のない高級言語では、さらに、並列処理のタイミングも分からないので、マトリクス状に配置された多数の演算ユニットを並列に動作させてアプリケーションを効果的に実行する設計は不可能である。一方、HDLは並列処理の記述は可能であるが、アルゴリズムを記述したとしても特定のハードウェアを前提としてアルゴリズムが記述できるだけなので、特定のハードウェアの知識が必要である。したがって、マトリクス状に配置された多種類の演算ユニットの機能および入出力のタイミングなどをソフトウェア技術者が理解し、アプリケーションをHDLで設計することは無理がある。

発明の開示

[0010] 本発明においては、並列に動作する複数の要素を備えたシステムにより、同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルを提供する。複数の並列記述は、他の並列処理の出力データが入力されるデータ入力を少なくとも含む複数のデータ入力を備えた第1の並列処理を示す第1の並列記述であって、それら複数のデータ入力には当該システムに入力されてからのレイテンシーが同一のデータが入力されることを示す第1の並列記述を含む。この定義ファイルにより、ハードウェア依存性のない記述により、プログラムカウンタの必要のない形態でアルゴリズムを定義することができる。すなわち、この定義ファイルにより、従来の高級言語と類似した記述ではあるが、時間順序性のない並列記述により、アルゴリズムに含まれる順序を時間的にではなく空間的に記述できる。

[0011] この定義ファイルは、処理させる仕事(アプリケーション)の手順を定義している点では、従来のプログラム言語と共通する。しかしながら、プログラム言語は、記述されたコンピュータに対する命令が、原則として最初から順番に、すなわち、時間的経過と共に実行されるものとして記述している。したがって、プログラム言語により定義されたアルゴリズムを実行するためには、実行する命令の順番を示すプログラムカウンタによる制御が必要となる。一方、プログラムに記述された命令はプログラムカウンタにより順番に実行されるので、変数を含む命令を実行する際は、その命令に時間的に先行して実行される命令により変数の状態は一義的に決まる。したがって、定義されたアルゴリズムは確実に実行され、実行できなくなるようなことは起こらない。

[0012] 本発明の定義ファイルは、同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する。したがって、本発明の定義ファイルに定義された並列処理を実行するには、時間的な順番を示すプログラムカウンタは必要としない。さらに、この定義ファイルでは、他の並列処理の出力データが入力される並列処理を示す並列記述を含むことによりアルゴリズムを定義できる。しかしながら、変数を含む並列記述では、変数の経過が一義的に定まらないとアルゴリズムが意図した通りに実行できなくなる可能性がある。そこで、複数のデータ入力を備えた第1の並列処理を示す第1の並列記述においては、それら複数のデータ入力には、当該システムに入

力されてからのレイテンシーが同一であるデータが入力されると解することにより、変数が不安定あるいは不確定になることを防止している。したがって、本発明の定義ファイルでは、並列記述毎に変数が一義的に定まり、アルゴリズムを並列記述により正確に定義できる。また、本発明の定義ファイルでは、並列記述にしたがって並列処理は完全に独立して実行され、同期通信などの余分な処理がなくても、また、ハードウェアを特定することによりタイミング問題を解決しなくても、アルゴリズムを正確に記述できる。

[0013] 本明細書において、入力されるデータのレイテンシーが同一とは、それらの入力されるデータが、同期してシステムにロードされたデータ(データ群)であるか、または、その同期してロードされたデータ(データ群)のいずれかであり、1つまたは複数の並列処理により処理されたデータであることを言う。すなわち、レイテンシーが同一のデータとは、システムに入力されたタイミングが同一のデータ、またはそれらのデータが他の並列処理により加工されたものである。

[0014] 本発明の定義ファイルは、HDLと同様に複数の並列記述からなり、複数の並列記述により並列処理を実行するためのプログラムカウンタは不要である。したがって、定義ファイルは、ハードウェアを記述していると言える。さらに、定義ファイルは、他の並列処理の出力データが入力されるデータ入力を備えた並列処理を示す並列記述を含み、複数のデータ入力がある場合もそれらに入力されるデータは一義的に決まる。したがって、本発明により、アルゴリズムを正確に、そして、ユーザが見て分かるように定義できるハードウェア記述言語を提供できる。さらに、ハードウェアの詳細が分からなくても各々の並列処理に入力されるデータは一義的に決まるので、実際に並列処理を行うハードウェアの詳細な情報あるいは知識は不要である。したがって、本発明の定義ファイルは、実際のハードウェアに依存せず、特定のハードウェアを前提にしないでハードウェアを記述できる。このため、ハードウェアインディペンデントで、極めて汎用的なハードウェア記述言語である。したがって、ソフトウェア技術者が簡単にLSIあるいはASIC、特に、並列処理要素を多数含んだLSIあるいはASICの設計あるいは生成するのに適したツールあるいは記述となる。

[0015] さらに、本発明の定義ファイルに含まれる複数の並列記述を、それらに入力される

データのレイテンシーの順番に並べると、複数の並列記述が、それに含まれる変数の時間経過の順番で並ぶことになる。この状態は、命令が実行される順番に並んだプログラムと同じである。したがって、ソフトウェア技術者は、普通にプログラムを作成するのと同じ感覚で、本発明の定義ファイルを作成することが可能である。この点でも、定義ファイルは、ソフトウェア技術者が簡単にLSIあるいはASICを設計あるいは生成するのに正に適したツールとなる。そして、本発明の定義ファイルにより、並列処理が可能な複数の要素により、アルゴリズムを空間的に割り付けることが可能となり、多数の並列に動作する要素を備えたシステムを有効に活用してアプリケーションを高速で実行できる。

[0016] 本発明の定義ファイルにおいては、並列記述毎に、入力データのレイテンシーが同一と判断され、タイミングが調整される。したがって、定義ファイルに従いハードウェアを生成する場合は、並列記述により規定される並列処理の単位で入力データのタイミングを調整する必要がある。並列処理を実行するための要素は、ビット単位で演算を行う演算ユニットであっても良い。ALUなどのある程度の規模の演算機能を備えた演算ユニットがマトリクス状に配置された並列処理システムにおいては、並列記述に対して1つまたは数個の演算ユニットを割り当てることにより並列処理を実行できる。したがって、並列記述単位でタイミングを調整するためには、遅延用の演算ユニットを加えれば良く、本発明の定義ファイルは、複数種類の演算ユニットを接続してアプリケーションに適したハードウェア構成を実現する並列処理システムを記述するのに適している。このため、本発明の定義ファイルにより、並列処理システムを効率的に設計および開発することができ、また、シミュレートすることが可能となる。

[0017] 本発明の定義ファイルは、コンピュータ読み取り可能な記録媒体に記録して提供することができ、定義ファイルに基づき、コンピュータを用いて、並列に動作する複数種類の要素を備えた並列処理システムを生成できる。すなわち、定義ファイルの複数の並列記述には、他の並列処理の出力データが入力されるデータ入力を少なくとも含む複数のデータ入力を備えた第1の並列処理を示す第1の並列記述が含まれている。この定義ファイルに従い、複数種類の要素の情報を記録したハードウェアライブラリに基づき、定義ファイルの並列記述に規定された並列処理を実行するための回路構

成(ハードウェア構成)であって、複数種類の要素の少なくともいずれかを備えた回路構成を含むハードウェア構成情報を生成する第1の工程と、第1の並列処理を実行するための回路構成の複数のデータ入力に、当該並列処理システムに入力されてからのレイテンシーが同一のデータが入力されるように、ハードウェア構成情報に遅延要素を加える第2の工程とを有する生成方法により、定義ファイルに定義されたアルゴリズムを複数種類の要素により空間的に割り付けるためのハードウェアを構成する情報を生成でき、アプリケーションを実行する並列処理システムを生成できる。

[0018] 特定のハードウェアに関する情報が格納されたハードウェアライブラリにより各種類の要素で消費される時間(サイクル数)が分かる。したがって、並列処理が複数のデータ入力を備えている場合に、ハードウェアライブラリを参照することにより、それら複数のデータ入力に入力されるデータのレイテンシーが一致するように、特定のハードウェアを前提とした遅延要素を加えることができ、特定のハードウェアに定義ファイルに定義されたアルゴリズムを正確に割付できる。複数の要素がクロックに同期して処理を実行するハードウェアであれば、ハードウェアライブラリには、複数種類の要素のそれぞれにおいて消費されるサイクル数を含む情報が格納されており、第2の工程では、複数種類の要素の少なくともいずれかにおいて消費されるサイクル数に相当する遅延要素が加えられる。

[0019] 並列処理システムは、ハードウェア構成が固定されるものであってもよい。並列処理システムは、複数種類の要素の接続を変えることにより異なるハードウェア構成を再構成可能なものであっても良い。再構成可能な並列処理システムに対しては、複数の異なるハードウェア構成を示す情報を備えているハードウェア構成情報が出力される。

[0020] 並列処理システムを生成する本発明の方法は、並列に動作する複数種類の要素を備えたシステムを生成する装置、例えばコンパイラとしても提供することが可能である。このコンパイラは、定義ファイルに基づき、並列に動作する複数種類の要素を備えた並列処理システムを生成する生成装置であって、複数種類の要素の情報を記録したハードウェアライブラリに基づき、定義ファイルの並列記述に規定された並列処理を実行するための回路構成であって、複数種類の要素の少なくともいずれかを備え

た回路構成を含むハードウェア構成情報を生成する第1の手段と、第1の並列処理を実行するための回路構成の複数のデータ入力に、当該並列処理システムに入力されてからのレイテンシーが同一のデータが入力されるように、ハードウェア構成情報に遅延要素を加える第2の手段とを有する。また、並列システムを設計するプロセスを適当なリソースを備えたコンピュータにより実現するためのプログラムあるいはプログラム製品として適当な記録媒体に記録したり、コンピュータネットワークを用いて提供することが可能である。この並列処理システムを設計するためのプログラムは、上述した第1の工程と、第2の工程と備えたプロセスをコンピュータにおいて実行可能な命令を有するプログラムである。

[0021] 本発明の定義ファイルは、複数種類の要素が、単体で1つの並列処理を処理可能な規模の複数種類の演算ユニットを備えた並列処理システムを生成するのに適している。したがって、ハードウェアライブラリには、単体で1つの並列処理を処理可能な程度の規模の複数種類の演算ユニットの情報を用意することが望ましい。また、複数種類の要素はビット単位で演算するものであっても良い。しかしながら、定義ファイルに記述される並列処理は一般にバイトあるいはワード単位のデータ処理である。したがって、ハードウェアライブラリには、バイトあるいはワード単位で異なった演算を実行可能な複数種類の演算ユニットの情報を用意し、複数の演算ユニットが配置された並列処理システムを生成することが望ましい。

[0022] 予め複数種類の演算ユニットがマトリクス状に配置され、演算ユニットを接続するネットワークあるいは回路配線の構成を変更することにより、異なるハードウェア構成を再構成可能な並列処理システムに対しては、コンパイラまたはコンパイラ用のプログラムにより、定義ファイルの内容を実行するのに適した、複数の異なるハードウェア構成を示す情報を備えたハードウェア構成情報を出力することができる。

[0023] 定義ファイルに基づいて並列処理システムを生成する際には、幾つかの最適化を施すことができる。複数の並列記述が、第3の並列記述により規定された第3の並列処理の少なくとも一部と同じ共通処理を含む第2の並列処理を規定する第2の並列記述を含んでいる場合は、第1の工程では、共通処理に対して、複数種類の要素の少なくともいずれかを含む共通の回路構成を生成し、第2の工程では、第2の並列処

理および共通処理の差分を実行するための回路構成を、第1の並列処理を実行するための回路構成として、遅延要素を加えることができる。第2の並列処理および共通処理の差分を実行するための回路構成は、第1の並列処理と同様に複数のデータ入力を備えたものとなるからである。これにより、回路を構成するための要素および、要素を繋ぐ配線などのハードウェアリソースの消費を抑制できる。

[0024] 並列処理システムを構成する複数種類の演算ユニットは、外部入力により処理を変更する手段を備えていても良い。それに対応して、複数の並列記述には、外部入力により処理が変更する並列処理を記述する並列記述を含められるようにすることが望ましい。ネットワークあるいは回路配線を変更しないでも処理内容を変更することができる並列処理システムの設計が可能となる。ネットワークあるいは回路配線と共に演算ユニットの処理内容を変更することも可能であり、さらにフレキシブルにリコンフィグラブルな並列処理システムの設計が可能となる。

[0025] 本発明の定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをシミュレートすることが可能である。定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをシミュレートする方法およびシミュレータは、定義ファイルに含まれる複数の並行処理を同期して実行する工程または手段を有し、この実行する工程では、複数の入力データを備えた並列記述を実行する際に、それら複数の入力データに、当該システムに入力されてからのレイテンシーが同一のデータを用いる。また、定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをコンピュータによりシミュレートするためのプログラムあるいはプログラム製品も本発明により提供され、CD−ROMなどの適当な記録媒体に記録したり、コンピュータネットワークを介して提供することができる。

図面の簡単な説明

[0026] [図1]図1は、定義ファイルであるDIDLからハードウェア構成情報であるDDDLを生成する概略構成を示す。

[図2]図2は、再構成可能な並列処理システムの概要を示す。

[図3]図3は、独立に並列に動作する複数のエレメントがマトリクス状に配置された並列処理システムを示す。

[図4]図4(a)は、DIDLの例を示し、図4(b)は、DIDLの異なる例を示す。

[図5]図5は、コンパイラの概略処理を示すフローチャートである。

[図6]図6は、コンパイラの概略構成を示すブロック図である。

[図7]図7(a)は、DIDLの例を示し、図7(b)は、それに対応する回路構成を示す。

[図8]図8(a)は、DIDLの異なる例を示し、図8(b)は、最適化された例を示し、図8(c)は、それに対応する回路構成を示す。

[図9]図9(a)は、DIDLのさらに異なる例を示し、図9(b)は、それに対応する回路構成を示す。

[図10]図10は、DIDLのさらに異なる例を示す。

[図11]図11は、図10に示すDIDLに対応する回路構成を示す。

[図12]図12は、図11に示す回路構成をマトリクスユニットに割り付けた状態を示す。

[図13]図13は、DIDLレベルでシミュレートする概略構成を示す。

[図14]図14は、DIDLレベルのシミュレータの処理の概要を示すフローチャートである。

発明を実施するための最良の形態

[0027] 図1に、本発明の定義ファイルを用いてハードウェアを設計する過程を示してある。定義ファイル1はDIDL(Device Independent Description Language)と称されており、コンパイラ2によりハードウェアライブラリ3の情報を参照して、ライブラリ3に格納されたハードウェアを用いたハードウェア構成情報4に変換される。ハードウェア構成情報4は、DDDL(Device Dependent Description Language)と称されている。コンパイラ2は適当なハードウェア資源を備えた汎用的なコンピュータ9を用いて実現されており、DIDL1を解釈してDDDL4を出力する並列システム設計用のプログラム5がインストールされることによりコンパイラとして機能する。したがって、DIDL1は、コンピュータ読み取り可能な記録媒体6、例えばCD-ROMあるいは磁気ディスクなどに記録されて提供される。インターネットなどのコンピュータネットワークによる通信を用いて提供することも可能であり、提供されたDIDL1は、コンピュータ9の一部となる適当な記録媒体に記録されて使用される。

[0028] 図2に、並列処理システムを備えたプロセッサの一例を示してある。このプロセッサ

は、本出願人の国際公開WO03／007155号に開示されている再構成可能なプロセッサ（RP、Reconfigurable Processor）である。このRP20は、プログラムなどによって与えられる命令セットに基づきエラー処理を含めた汎用的な処理を行う汎用的な構成の基本プロセッサ21と、マトリクス状に配置された演算あるいは論理エレメントにより特定のデータ処理に適合した1つまたは複数のデータパス（データフローあるいは擬似データフロー）がバリアブルに形成されるAAP（Adaptive Application Processor）ユニット（以降ではAAP）50と、このAAP50からの割り込み処理を制御する割り込み制御ユニット22と、AAP50に作動用のクロック信号を供給するクロック発生部28と、このRP20で提供可能な演算回路のフレキシビリティーをさらに向上するためのFPGAユニット27と、外部に対するデータの入出力を制御するバス制御ユニット29とを備えている。基本プロセッサ21とAAP50は、これらの間でデータを交換可能なデータバス24aと、基本プロセッサ21からAAP50の構成および動作を制御するための命令バス24bとにより接続されている。また、AAP50から割り込み制御ユニット22に信号線25を介して割り込み信号が供給され、AAP50における処理が終了したり、処理中にエラーが発生したときはAAP50の状態を基本プロセッサ21にフィードバックできるようになっている。

[0029] AAP50とFPGA27との間もデータバス26により接続されており、AAP50からFPGA27にデータを供給して処理を行い、その結果をAAP50に返せるようになっている。さらに、AAP50は、ロードバス23aおよびストアバス23bによってバス制御ユニット29と接続されており、RP20の外部のデータバスとの間でデータを交換できるようになっている。基本プロセッサ21もバス21aによりバス制御ユニット29と接続されており、外部のデータパスとの間でデータを交換できる。

[0030] 図3にAAP50の概要を示してある。AAP50は、複数の算術および／または論理演算を行う論理要素（論理ブロックあるいは論理ユニット（以降ではエレメントと称する））がマトリクス状に配置されたマトリクス部51と、そのマトリクス部51に対してデータを供給する入力バッファ52aと、マトリクス部51から出力されるデータを格納する出力バッファ52bを備えている。これら入力バッファ52aおよび出力バッファ52bは、それぞれ4つの小容量の入力メモリにより構成されており、アクセス調停ユニット54を介して

入出力バス23aおよび23bに接続される。

[0031] このマトリクス部51が、データパスあるいはデータフローを再構成可能な並列処理システムの中心となる集積回路区画であり、並列に動作する複数種類の演算ユニットであるエレメント55が縦方向に4つのラインを構成するようにアレイ状あるいはマトリクス状に配置されている。そして、マトリクス51に含まれている複数種類のエレメントの情報がハードウェアライブラリ3に格納されている。このマトリクス部51は、これらのエレメント55の間に配置された、横方向に延びた行配線群57と、縦方向に延びた列配線群58とを備えている。列配線群58は、列方向に並んだ演算ユニット55の左右に分かれて配置された配線群58xおよび58yが1対になっている。行配線群57および列配線群58との交点にはスイッチングユニット59が配置されており、行配線群57の任意のチャンネルを、列配線群58の任意のチャンネルに切り替えて接続できるようになっている。各々のスイッチングユニット59は、設定を記憶するコンフィグレイションRAMを備えており、プロセッサ部21から供給されるデータによりコンフィグレイションRAMの内容を書き換えることにより、行配線群57と列配線群58の接続を動的に任意に制御できる。このため、このマトリクス部51においては、複数のエレメント55の全部あるいは一部が配線群57および58により接続されて形成されるデータフローの構成を任意に動的に変更することができる。

[0032] RP20においては、これらのエレメント55が並列に動作し、各種類のエレメント55の機能、遅延、入出力データの条件などの情報がハードウェアライブラリ3に格納されている。また、これらのエレメント55は、クロック発生部28から供給されるクロック信号に同期して稼動するので、種類あるいはエレメント内部で実行される処理により消費されるサイクル数(クロック数)が決まる。したがって、ハードウェアライブラリ3には各種類のエレメント毎に、入力データを処理して出力するために消費されるサイクル数が遅延情報として格納されている。さらに、各種類のエレメント55の配置と、配線群57および58、スイッチングユニット59の情報もハードウェアライブラリ3に格納されており、コンパイラ2からは、DIDL1に定義されたアルゴリズムを実現するための、エレメント55の接続情報(データフロー構成)がハードウェア構成情報(DDDL)4として出力される。このため、DDDL4に従ってエレメント55が配線群57および58で接続されるよ

うにマトリクス部51を制御することにより、DIDL1に定義されたアルゴリズムをマトリクス部51のエレメント55により空間的に割り付けることが可能となる。

[0033] 各エレメント55は、1組の列配線群58xおよび58yのそれぞれから入力データを選択するための1組のセレクタ53と、選択された入力データに特定の算術および／または論理演算処理を施し、出力データとして行配線群57に出力する内部データパス部56を備えている。そして、本例のマトリクス部51には、各行毎に異なる処理を行うための内部データパス部56を備えた種類の異なるエレメント55が並んで配置されている。例えば、第1行目に配列されたエレメント55は、入力バッファ52aからのデータを受信する処理に適したデータパス部（LD）56iを備えている。第2行目に配置されたエレメント55aは、入力バッファ52aに外部デバイスからデータを書き込むためのエレメントであり、ブロックロードするためのアドレスを発生するのに適した内部データパスを具備するデータパス部（BLA）56aを備えている。マトリクス51を構成する全てのエレメント55は、内部データパスの構成あるいは初期値などがある程度変更できるようになっている。その設定は、各々のエレメント55のコンフィグレイションRAMに、制御バス24bを介して基本プロセッサ21から制御信号により指示される。

[0034] 第3行目に配置されたエレメント55bは、入力RAMの各々より所望のデータをマトリクス部51へロードする入力読み出しアドレスを発生するデータパス部（LDA）56bを備えている。第4行目および第5行目に配列されたエレメント55cは、算術演算および論理演算に適したデータパス部（SMA）56cを備えている。このデータパス部56cは、たとえば、シフト回路、マスク回路、論理演算ユニットALUおよびALUで処理する演算をセットするコンフィグレイションRAMを備えている。したがって、プロセッサ21が書き込んだ命令により、マトリクス部51へ入力されたデータを加算あるいは減算したり、比較したり、論理和あるいは論理積を取ったりすることができ、その結果がエレメント55の出力信号として出力される。

[0035] その下の行に配列されたエレメント55dは、データが伝送されるタイミングを遅延する処理に適したデータパス部（DEL）56dを備えている。その下の行に配列されたエレメント55eは、乗算器などを含む乗算処理に適したデータパス部（MUL）56eを備えている。さらに異なるエレメント55fとしては、マトリクス部51の外部に用意されたFP

GA27とのインターフェイス用のデータパス部56fを備えたエレメントも用意されており、データをいったんFPGA27に供給して処理した後、再びマトリクス部51に戻して処理を継続することができる。

[0036] 再構成可能な集積回路区画であるマトリクス部51には、さらに、ストア用のアドレスを発生するのに適したデータパス部56gおよび56hをそれぞれ備えたエレメント55gおよび55hが配置されている。これらは、出力バッファ52bを介して外部デバイスにデータを出力するための制御を行う。そして、最下段には、ストア用にデータを出力するのに適したデータパス部(ST)56sを備えたエレメント55が配列されている。したがって、マトリクス部51を用いて、エレメント55の接続を動的に変更することにより、様々なデータフローをフレキシブルに構成でき、様々な処理を行うことができる。

[0037] 図4(a)および図4(b)に、DIDLの簡単な例を示してある。図4(a)に示したDIDL10aは2行の並列記述11aおよび11bを有している。並列記述11aは、変数aに変数bを代入する処理12aを規定している。また、並列記述11bは、変数cに変数aを代入する処理12bを規定している。したがって、処理12bは、他の処理12aの出力データをデータ入力とする処理となる。これらの処理12aおよび12bは、並列に動作する要素により同期して独立に行われる並列処理であり、DIDL10aはそれらの並列処理12aおよび12bを同期して独立に実行するハードウェア構成を示している。このDIDL10aにより定義されたハードウェアにおいては、あるサイクルt0で変数(a、b、c)が(1、2、3)であれば、次のサイクルt1で変数(a、b、c)は(2、2、1)となる。

[0038] 一方、このDIDL10aの記載がプログラム19aだと理解すると、変数cは2になるので、得られる結果は異なる。しかしながら、サイクルt1に続く次のサイクルt2において、DIDL10aにより定義されたハードウェアにおいては、変数(a、b、c)は(2、2、2)となり、DIDL10aの記載がプログラム19aであると理解した場合と同じ結果が得られる。

[0039] 図4(b)に示したDIDL10a´は、並列記述11aおよび11bの順番が入れ替わっているが、それぞれの記述に対応する処理12aおよび12bが独立して行われるので各サイクルt0ーt2における演算結果は変わりない。これに対し、図4(b)の記述がプログラム19bであると理解すると、処理12aと12bの順番が入れ替わるので、変数(a、b、c)は(2、2、1)となる。すなわち、処理が入れ替わったプログラムの結果は、DIDL

10aまたは10bのサイクルt1における値に合致する。

[0040] このように、DIDL10aにより、プログラム19aおよび19bのいずれのアルゴリズムも実現するハードウェアを記述できるが、プログラム19aおよび19bと同一の結果が得られるサイクルが異なる。そこで、本発明においては、DIDL10aを、少なくとも1つの他の並列処理の出力データを少なくとも1つのデータ入力とする並列処理を示す並列記述によりアルゴリズムを並列システムに割り付ける定義ファイルとし、さらに、複数のデータ入力を備えた並列処理(第1の並列処理)を示す並列記述(第1の並列記述)においては、それら複数のデータ入力には、システム、すなわちマトリクス51に入力されてからのレイテンシーが同一であるデータが入力されると規定することにより、プログラムと同一のアルゴリズムにより演算が可能なハードウェア構成を定義できるようにしている。

[0041] 図5に、DIDL用のコンパイラ9の概略処理を示してある。まず、ステップ31でDIDL1を読み込み、ステップ32でDIDL1に含まれた並列記述を解釈する。そして、ハードウェアライブラリ3に格納されたハードウェア構成に基づき、並列記述に示された並列処理を行う回路構成を生成する。図3に示した再構成可能なプロセッサのマトリクス部51により並列処理を行うハードウェア構成情報4を生成するために、ハードウェアライブラリ3に格納された、各種のエレメント55の情報と、配線57および58とスイッチ59の情報とを用い、エレメント55を備え、並列に動作する回路構成を生成し、ハードウェア構成情報のDDDL4としてコンパイラ2の適当なメモリに格納する。各エレメント55の情報としては、例えば、演算機能、入力条件、処理サイクルがある。マトリクス部51では、各エレメント55の個数および配置は決まっているので、並列処理を行う回路構成のハードウェア情報4としては、並列処理のために選択されたエレメント55と、その位置および選択されたエレメント55を接続する配線ルートといった回路情報を含むものが生成される。

[0042] ステップ32において生成された回路構成が複数のデータ入力を備えている場合は、ステップ33において、それら複数のデータ入力に、並列処理システム、すなわちマトリクス部51に入力されてからのレイテンシーが同一のデータが入力されるように、遅延要素となるデータパス部(DEL)56dを備えたエレメント55dの情報をDDDL4に加

える。したがって、DIDL1に、他の並列処理の出力データが入力されるデータ入力を少なくとも含む複数のデータ入力を備えた第1の並列処理を示す第1の並列記述が含まれている場合は、第1の並列処理を実行するためにステップ32において生成された回路構成の複数のデータ入力に、並列処理システムに入力されてからのレイテンシーが同一のデータが入力されるように、ステップ33において、ハードウェア構成情報であるDDDL4に遅延要素を加える。

[0043] さらに、ステップ32および33では、幾つかの最適化が行われる。DIDL1に含まれる並列記述により規定された並列処理(第2の並列処理)が、他の並列記述により規定された並列処理(第3の並列処理)の少なくとも一部と同じ共通処理を含む場合は、ステップ32では、共通処理に同一のエレメント55を含む共通な回路構成を割り当て、ステップ33では、共通構成に含まれないエレメント55、すなわち、第2の並列処理および共通処理の差分を実行するための回路構成に対して、必要であれば遅延エレメント55dを加えてレイテンシーを調整する。

[0044] そして、ステップ34において、ステップ32および33で生成された回路構成とそれらを接続する情報を含むDDDL4を出力する。マトリクス部51は、複数のエレメント55の接続を変えることにより異なるハードウェア構成を再構成可能となっている。このため、DDDL4は、複数の異なるハードウェア構成を示す情報となっており、DDDL4によりマトリクス部51を再構成することができる。これらのステップは、DIDL1に含まれる並列記述を個別に読み込んで行うことも可能であり、DIDL1に含まれる全てあるいは一部の並列記述を読み込んで行うことも可能である。

[0045] これらのステップ31ー34を含む並列処理システムの生成方法は、ステップ31ー34の各工程を汎用コンピュータ9で実行可能なコンパイラプログラムあるいはプログラム製品5として適当な記録媒体に記録して提供することができる。また、ネットワークを介してプログラム製品を提供することも可能である。そして、汎用コンピュータ9にプログラム5をインストールすることにより汎用コンピュータをコンパイラ2として使用できる。したがって、コンパイラプログラム5をインストールされたコンピュータ9は、図6に示すように、DIDL1を読み込む機能35と、ハードウェアライブラリ3に基づき、DIDL1に記述された並列処理を実行するための回路構成を生成する機能(第1の手段)36と、

回路構成が複数のデータ入力を備えている場合は、それら複数のデータ入力に対し、マトリクス部51に入力されてからのレイテンシーが一致するデータが供給されるように遅延エレメント55dを加える機能(第2の手段)37と、生成された回路構成を、エレメント間の接続情報を含めてDDDL4として出力する機能38を備えたコンパイラ2として動作する。

[0046] 図7(a)に、異なるDIDLの例を示してある。このDIDL10bには、システム、本例ではマトリクス部51に入力される変数を示す記述11cと、内部変数を示す記述11dと、足し算を示す記述11eとが含まれている。このDIDL10bがコンパイラ2に読み込まれて処理されると、図7(b)に示すように、算術演算が可能なデータパス56cを備えた演算エレメント55cを有する回路構成18bが生成される。この回路構成18bは、演算エレメント55cに、変数bとcが入力される2つのデータ入力と、変数aが出力されるデータ出力とを備えている。

[0047] 図8(a)に、さらに異なるDIDLの例を示してある。このDIDL10cには、変数に関する記述11cおよび11dに加え、2つの並列処理12fおよび12gを示す並列記述11fおよび11gが含まれている。このDIDL10cがコンパイラ2に読み込まれると、並列処理12fおよび12gは共通した処理を含んでいるので、ステップ32において、共通の部分に対して共通の回路構成17cが生成される。すなわち、DIDL10cは、図8(b)に示すように、並列処理12gが並列処理12g´に最適化される。

[0048] 差分の処理を行う並列処理12g´は、他の並列処理12fの出力データ「a」が入力されるデータ入力と、他の並列処理12fを経ないデータ「d」が入力されるデータ入力とを含む複数のデータ入力を備えた第1の並列処理となる。したがって、並列処理12g´を実行する回路構成に含まれる演算エレメント55cは、変数aが供給される入力と、変数dが供給される入力とを備えている。変数aは、マトリクス部51に入力される変数bおよびcに対し、演算エレメント55cの処理サイクル分だけ遅れる変数である。また、変数dは、他の変数bおよびcと同時にマトリクス51に入力される変数である。このため、ステップ33においては、並列処理12g´を行うエレメント55cに入力される変数aと変数dとのレイテンシーを調整するように、マトリクス部51に入力される変数dを、並列処理12fを行う演算エレメント55cのサイクル数分だけ遅らせる遅延エレメント55dが挿

入される。

[0049] 演算エレメント55cで足し算を行った場合に消費されるサイクル数は、ハードウェアライブラリ3に格納されている。このため、ステップ33においては、ハードウェアライブラリ3に格納されている情報に基づき、足し算のエレメントにおいて消費されるサイクル数に相当するサイクルを消費する遅延エレメント55dがDDDL4に追加される。その結果、DDDL4は、図8(c)に示す回路構成18cを含むハードウェア構成情報を含んで生成され、ステップ34においてコンパイラ2から出力される。なお、以降では、説明を簡単にするために、特に記載しないかぎり、エレメントにおいては1サイクルで処理されるものとして説明する。

[0050] 図9(a)に、さらに異なるDIDLの例を示してある。このDIDL10dには、変数に関する記述11cおよび11dに加え、4つの並列処理12h一12kを示す並列記述11h一11kが含まれている。このDIDL10cがコンパイラ2に読み込まれると、まずは、回路構成を生成するステップ32において、並列処理12h一12kを実行するための演算エレメント55cを用いた回路構成が生成される。並列処理12kに入力される1つの変数cは、マトリクス51に入力された変数(以降ではシステム入力変数)aを入力とする並列処理12hの出力である。また、並列処理12kに入力される他の変数eは、システム入力変数aおよびbを入力とする並列処理12iの出力をさらに入力とする並列処理12jの出力である。したがって、レイテンシーを調整する工程33においては、並列処理12kに入力される変数cと変数eとのレイテンシーを調整するために、変数eに対して変数cを遅らせて最後の足し算を行うエレメント55cのデータ入力に供給するための遅延エレメント55dが追加される。その結果、図9(b)に示す回路構成18dが生成され、複数のエレメント55を接続するハードウェア構成情報、すなわちDDDL4としてコンパイラ2から出力される。

[0051] 図9(a)の記述をプログラム19dとして考えた場合、処理12h一12kは時間軸に従って上から順番に行われる。したがって、処理12kで演算される変数cおよび変数eは、それぞれ、先行する処理12hおよび12jによりそれぞれ決定されたものとなる。一方、図9(a)の記述を単に並列実行される複数の処理の記述であると考えると、並列処理12kの変数cおよび変数eは、1サイクル前に定まった値となり、プログラム19dと

して考えた場合と処理12kの出力が異なる。これに対し、図9(a)の記述が本発明の定義ファイル、すなわちDIDL10dであるとすると、並列処理12kの変数cおよび変数eは、レイテンシーが同一であると解釈される。したがって、2サイクル前に確定した変数cが遅延エレメント55dを介して1サイクル前に確定した変数eと同期を取って並列処理12kに入力される。その結果、並列処理12kの出力fは、変数aおよびbを入力してから3サイクル後にプログラム19dと同じになり、並列処理を記述したDIDL10dによりプログラム19dと同じ結果が得られる。

[0052] 並列処理を記述したDIDL10dは、それに含まれる記述11h一11kの順番を変えても処理される内容は同じである。これに対し、プログラム19dは、処理がならんだ順番、すなわち、時系列でアルゴリズムを定義しているので、記述を入れ替えると処理される内容は異なる。しかしながら、複数の並列記述に対して入力変数のレイテンシーが同一になるという定義を導入することにより、プログラムとして記述されたファイルを、並列処理を定義するファイル、すなわち、ハードウェアを定義するファイルとして理解することが可能となる。この結果、プログラマーは、普通に時系列に従ってアルゴリズムを示したプログラムを記述することにより、並列処理システムのハードウェアを記述する本発明の定義ファイルを作成することが可能となる。したがって、本発明の定義ファイルであるDIDLを用いることにより、プログラマーはプログラムを作成するのと同じ感覚でハードウェア設計を行うことができる。

[0053] さらに、定義ファイルでは、並列処理に入力される変数のレイテンシーが同一と理解されるだけであり、特定のハードウェアを前提としない。すなわち、レイテンシーが同一の入力変数は、同期して、あるシステム(本例ではマトリクスユニット51)にロードされたデータ群に含まれるものか、またはその同期してロードされたデータ群のいずれかが、他の並列処理により処理されたものである、同期してシステムに入力された変数由来であること以外は、実際のハードウェアを前提として設計するとき以外に理解する必要がない。したがって、本発明の定義ファイルは、ハードウェアに依存しないハードウェア記述言語であると言うことができる。すなわち、どのようなハードウェアを前提としても解釈することができ、ハードウェアが特定されれば、そのハードウェアに定義ファイルに記述されたアルゴリズムを割り付けることができる。したがって、本例

のDIDLのような定義ファイルは、時間的ではなく、空間的にアルゴリズムを展開できる言語ということができる。このため、プログラムで記述できるアルゴリズムは、本例のDIDLを用いることにより、すべて、並列に動作する複数の要素、本例であればエレメント55を組み合わせた回路により実行させることができ、DIDLにより、実行するためのハードウェア構成を記述することができる。

[0054] 図10に、さらに異なるDIDLの例を示してある。このDIDL10eには、外部から与えられる所定の数(num0fData)の入力変数inの最大値aを検索する処理が記述されている。初期値をセットする記述111に続く部分が並列処理を記述した部分であり、入力変数inとそれまでの最大値aとを比較して最大値をセットする並列処理12mの並列記述11mと、カウンターを進める並列処理12nの並列記述11nと、処理12nのカウントアップした値が所定の数num0fDataに達したら処理12mの最大値aを出力する並列処理12oの並列記述11oとを備えている。処理12oでは、上記の処理12mおよび12nに要するサイクル数の差を吸収するために、遅延素子を挿入して2つのデータ入力に供給される変数のレイテンシーが調整される。

[0055] 図11に、このDIDL10eがコンパイルされたハードウェア構成18eを模式的に示してある。回路構成を生成するステップ32において、並列処理12mは、演算エレメント55cを用いて回路構成され、並列処理12nは、アドレス生成用のエレメント55bを用いて回路構成され、並列処理12oは2つの演算エレメント55cを用いて回路構成され、それらの回路構成を示すDDDL4が生成される。処理12mを行う演算エレメント55cにおいて消費されるサイクル数が、処理12nのカウント処理を行うエレメント55bにおいて消費されるサイクル数より多い。このため、遅延エレメント55dが並列処理12oのカウンター値を入力する側に追加される。これにより、並列処理12oのデータ入力に供給されるデータ、すなわち、処理12mの出力と処理12nの出力のレイテンシーが調整される。なお、処理12oの出力側に配置された2つの遅延エレメント55dは、マトリクス部51が3つのセグメントに別れており、最初のセグメントで処理12m－12oの回路が構成されるので、他のセグメントを通過して出力用のエレメント56sにデータを転送するためのものである。

[0056] 図12は、図3に示したマトリクス部51に配置されたエレメント55に図11に示した回

路構成を割り付けた状態が示されている。本例においては、エレメント55が既にマトリクス状に配置された再構成可能なプロセッサ20のマトリクス部51に、DIDL10eのアルゴリズムを割り当てている。したがって、図11に示したハードウェア構成の情報は、図12に示したように、選択されたエレメントの配置と、それらを接続する配線ルートの情報としてコンパイラ2で生成され、DDDL4として出力される。なお、アドレスを生成するエレメント55aおよび55bは、入力変数inを外部から入力するために使用されている。また、本図には示していないが、出力用のアドレスを生成するエレメント55gおよび55hも同様に使用されている。

[0057] 本例の並列処理システム20は、エレメント55がマトリクス状に配置された再構成可能なプロセッサ(集積回路装置)であり、各エレメント55は、上述したように、ALUなどのある程度の規模の演算機能を備えており、1つのエレメントにより1つの並列処理をほぼ実行できる規模となっている。また、各エレメント55は、データを8ビット、16ビットあるいは32ビットなどのバイトあるいはワード単位で、特定の目的のために処理するのに適した演算ユニットである。そして、入力データと出力データとをフリップフロップなどを使ってラッチし、クロック信号で同期している。すなわち、各エレメント55における入力と出力はクロックで同期されている。したがって、各エレメント55で消費されるサイクルは予め予想することができる。さらに、各エレメントは特定の処理を行うのに適したある程度の規模の演算機能を備えているので、DIDL1に記述された並列処理にエレメントの単位でハードウェアを割り当てて回路構成することができる。したがって、本例の並列処理システムであるマトリクスユニット51を前提としてDIDL1を解釈すると並列処理で消費されるサイクルは容易に予想でき、回路構成を生成するのが容易であり、さらにレイテンシー調整も容易に行える。このため、ある程度の規模の演算能力を備えて、1つの並列処理をエレメント単位でほぼ実行できるマトリクス部51を備えたデータ処理装置であるRP20は、本発明の定義ファイルからハードウェアを生成するのに適したアーキテクチャであると言える。

[0058] また、エレメント55の単位で並列処理が割り付けられ、サイクル数も管理されるので、DIDL1で定義されたアプリケーションを実行している途中で、あるエレメント55の機能を外部入力により変えても、他の並列処理に予期しない影響を及ぼさずにアプリケ

ーションを実行させることができる。機能を変えたエレメント55におけるサイクル数が変動するのであれば、それに対応できるように遅延エレメント55dを接続しておき、遅延エレメント55dの内部のサイクル数を同様に外部入力により変更し、他のエレメント55で形成されたデータフローに影響を及ぼさずに特定のエレメント55の処理をダイナミックに変更することが可能である。

[0059] 図13に、DIDLレベルのシミュレータ67を示してある。このシミュレータ67は汎用のコンピュータ9にシミュレータ用のプログラム68をインストールすることにより構成される。したがって、本発明の定義ファイルであるDIDL1に基づき、並列に動作する複数種類の要素を備えたシステムをコンピュータ9によりシミュレートするためのプログラムあるいはプログラム製品68も、CD−ROMなどの適当な記録媒体に記録したり、コンピュータネットワークを介して提供することができる。

[0060] 図14にシミュレータ67の概略動作をフローチャートにより示してある。まず、ステップ71でDIDL1を読み込む。次に、ステップ72で、DIDL1に記述された複数の並列処理を同期して実行する。この際、第1の並列記述に示された、複数のデータ入力を備えた第1の並列処理については、それら複数のデータ入力に、システムに入力されてからのレイテンシーが同一のデータを入力する。例えば、システムに入力され、他の並列処理で加工されないデータとして定義されているデータ(システム入力変数)に対するレイテンシーが同一になるデータを用いる。ステップ73で終了条件、例えば、所定の回数、並列処理を繰り返して実行したり、DIDL1に記述された並列処理の結果が所定の値に達するなどの条件が成立すると、ステップ74でシミュレーションした結果を出力する。これにより、ハードウェアに依存しないで、DIDL1に記述されたハードウェアの動作をシミュレートすることができる。

[0061] 以上に説明したように、本発明においては、同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルであって、複数のデータ入力を備えた並列記述においては、それら複数のデータ入力に対してシステムに入力されてからのレイテンシーが同一のデータが供給されると解釈する定義ファイルを提案している。この定義ファイルは、並列処理を記述するものであり、ハードウェア記述ファイルであると理解することができ、また、そこにはハードウェア自体は表れ

ないのでハードウェア依存性のないハードウェア記述であると言うことができる。さらに、並列記述であるので、プログラムカウンタの必要のない形態でアルゴリズムを定義することができる。

[0062] したがって、本発明の定義ファイルにより、従来の高級言語と類似した記述ではあるが、時間順序性のない並列記述により、アルゴリズムに含まれる順序を時間的にではなく空間的に記述でき、並列に動作する複数の要素を備えた並列処理システムの生成を短期間で容易に行うことが可能となる。特に、ALUなどのある程度の規模の演算機能を備えた演算ユニットがマトリクス状に配置された並列処理システム、さらには、演算ユニットの接続を変えられる再構成可能なデータ処理システムを設計したり開発したりするのに本発明の定義ファイルは有用である。

請求の範囲

[1] 定義ファイルに従い、並列に動作する複数種類の要素を備えた並列処理システムを生成する方法であって、

前記定義ファイルは、独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有し、前記複数の並列記述は、他の並列処理の出力データが入力されるデータ入力を少なくとも含む複数のデータ入力を備えた第1の並列処理を示す第1の並列記述を含んでおり、

前記複数種類の要素の情報を記録したハードウェアライブラリに基づき、前記定義ファイルの並列記述に規定された並列処理を実行するための回路構成であって、前記複数種類の要素の少なくともいずれかを備えた回路構成を含むハードウェア構成情報を生成する第1の工程と、

前記第1の並列処理を実行するための回路構成の複数のデータ入力に、当該並列処理システムに入力されてからのレイテンシーが同一のデータが入力されるように、前記ハードウェア構成情報に遅延要素を加える第2の工程とを有する、生成方法。

[2] 前記並列処理システムは、前記複数種類の要素の接続を変えることにより異なるハードウェア構成を再構成可能であり、前記ハードウェア構成情報は、複数の前記異なるハードウェア構成を示す情報を備えている、請求項1の生成方法。

[3] 前記複数種類の要素は、単体で前記定義ファイルの1つの並列記述に規定された並列処理を処理可能な規模の複数種類の演算ユニットを含んでいる、請求項1の生成方法。

[4] 前記複数種類の要素は、バイトあるいはワード単位で異なった演算を実行可能な複数種類の演算ユニットを含んでいる、請求項1の生成方法。

[5] 前記ハードウェアライブラリには、前記複数種類の要素のそれぞれにおいて消費されるサイクル数を含む情報が格納されており、

前記第2の工程では、前記複数種類の要素の少なくともいずれかにおいて消費されるサイクル数に相当する前記遅延要素を加える、請求項1の生成方法。

[6] 前記複数の並列記述は、第3の並列記述により規定された第3の並列処理の少なくとも一部と同じ共通処理を含む第2の並列処理を規定する第2の並列記述を含んで

おり、

前記第1の工程では、前記共通処理に対して、前記複数種類の要素の少なくともいずれかを含む共通の回路構成を生成し、

前記第2の工程では、前記第2の並列処理および前記共通処理の差分を実行するための回路構成を、前記第1の並列処理を実行するための回路構成として、前記遅延要素を加える、請求項1の生成方法。

[7] 定義ファイルに従い、並列に動作する複数種類の要素を備えた並列処理システムを生成する装置であって、

前記定義ファイルは、独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有し、前記複数の並列記述は、他の並列処理の出力データが入力されるデータ入力を少なくとも含む複数のデータ入力を備えた第1の並列処理を示す第1の並列記述を含んでおり、

前記複数種類の要素の情報を記録したハードウェアライブラリに基づき、前記定義ファイルの並列記述に規定された並列処理を実行するための回路構成であって、前記複数種類の要素の少なくともいずれかを備えた回路構成を含むハードウェア構成情報を生成する第1の手段と、

前記第1の並列処理を実行するための回路構成の複数のデータ入力に、当該並列処理システムに入力されてからのレイテンシーが同一のデータが入力されるように、前記ハードウェア構成情報に遅延要素を加える第2の手段とを有する、生成装置。

[8] 前記並列処理システムは、前記複数種類の要素の接続を変えることにより異なるハードウェア構成を再構成可能であり、前記ハードウェア構成情報は、複数の前記異なるハードウェア構成を示す情報を備えている、請求項7の生成装置。

[9] 定義ファイルに従い、並列に動作する複数種類の要素を備えたシステムを設計するプロセスをコンピュータにより実行するためのプログラムであって、

前記定義ファイルは、独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有し、前記複数の並列記述は、他の並列処理の出力データが入力されるデータ入力を少なくとも含む複数のデータ入力を備えた第1の並列処理を示す第1の並列記述を含んでおり、

前記システムを設計するプロセスは、前記複数種類の要素の情報を記録したハードウェアライブラリに基づき、前記定義ファイルの並列記述に規定された並列処理を実行するための回路構成であって、前記複数種類の要素の少なくともいずれかを備えた回路構成を含むハードウェア構成情報を生成する第1の工程と、

前記第1の並列処理を実行するための回路構成の複数のデータ入力に、当該並列処理システムに入力されてからのレイテンシーが同一のデータが入力されるように、前記ハードウェア構成情報に遅延要素を加える第2の工程とを有する、プログラム。

[10] 並列に動作する複数の要素を備えたシステムにより、同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルであって、前記複数の並列記述は、他の並列処理の出力データが入力されるデータ入力を少なくとも含む複数のデータ入力を備えた第1の並列処理を示す第1の並列記述であって、それら複数のデータ入力には当該システムに入力されてからのレイテンシーが同一のデータが入力されることを示す第1の並列記述を含む、定義ファイルが記録されているコンピュータ読み取り可能な記録媒体。

[11] 前記複数の並列記述は、前記複数の要素が動作するクロックに同期して実行される前記複数の並列処理をそれぞれ規定する、請求項10の記録媒体。

[12] 定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをシミュレートする方法であって、

前記定義ファイルは、独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有し、前記複数の並列記述は、他の並列処理の出力データが入力されるデータ入力を少なくとも含む複数のデータ入力を備えた第1の並列処理を示す第1の並列記述を含んでおり、

前記定義ファイルに規定された前記複数の並行処理を同期して実行する工程を有し、この工程では、前記第1の並列処理の複数のデータ入力に、当該システムに入力されてからのレイテンシーが同一のデータを入力する、シミュレーション方法。

[13] 定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをシミュレートするシミュレータであって、

前記定義ファイルは、独立に行われる複数の並列処理をそれぞれ規定した複数の

並列記述を有し、前記複数の並列記述は、他の並列処理の出力データが入力されるデータ入力を少なくとも含む複数のデータ入力を備えた第1の並列処理を示す第1の並列記述を含んでおり、

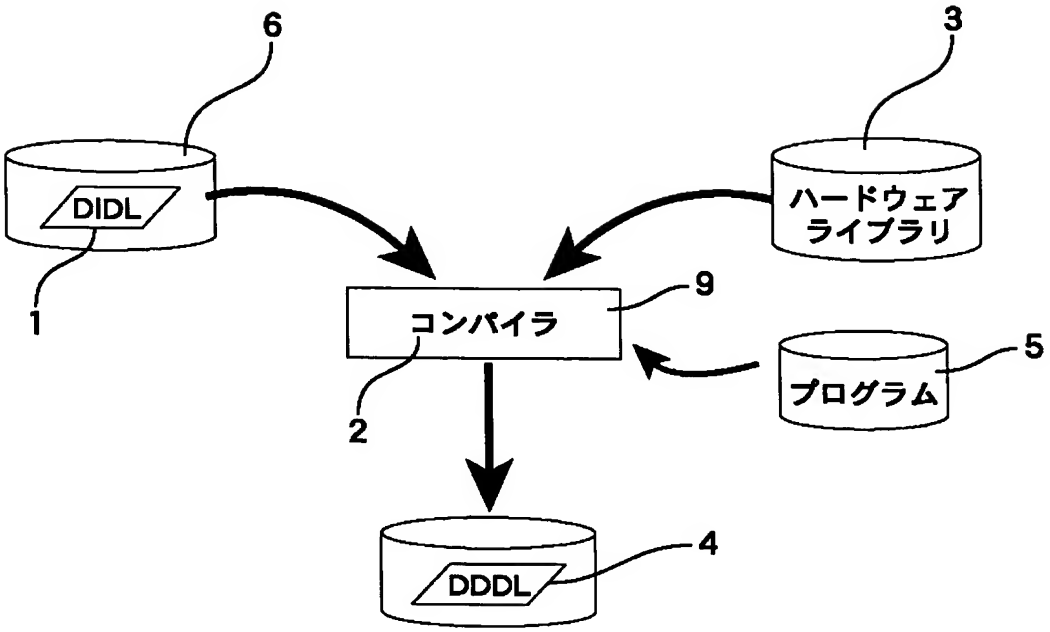
前記定義ファイルに規定された前記複数の並行処理を同期して実行する手段を有し、この実行する手段では、前記第1の並列処理の複数のデータ入力に、当該システムに入力されてからのレイテンシーが同一のデータを入力する、シミュレータ。

[14] 定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをコンピュータによりシミュレートするためのプログラムであって、

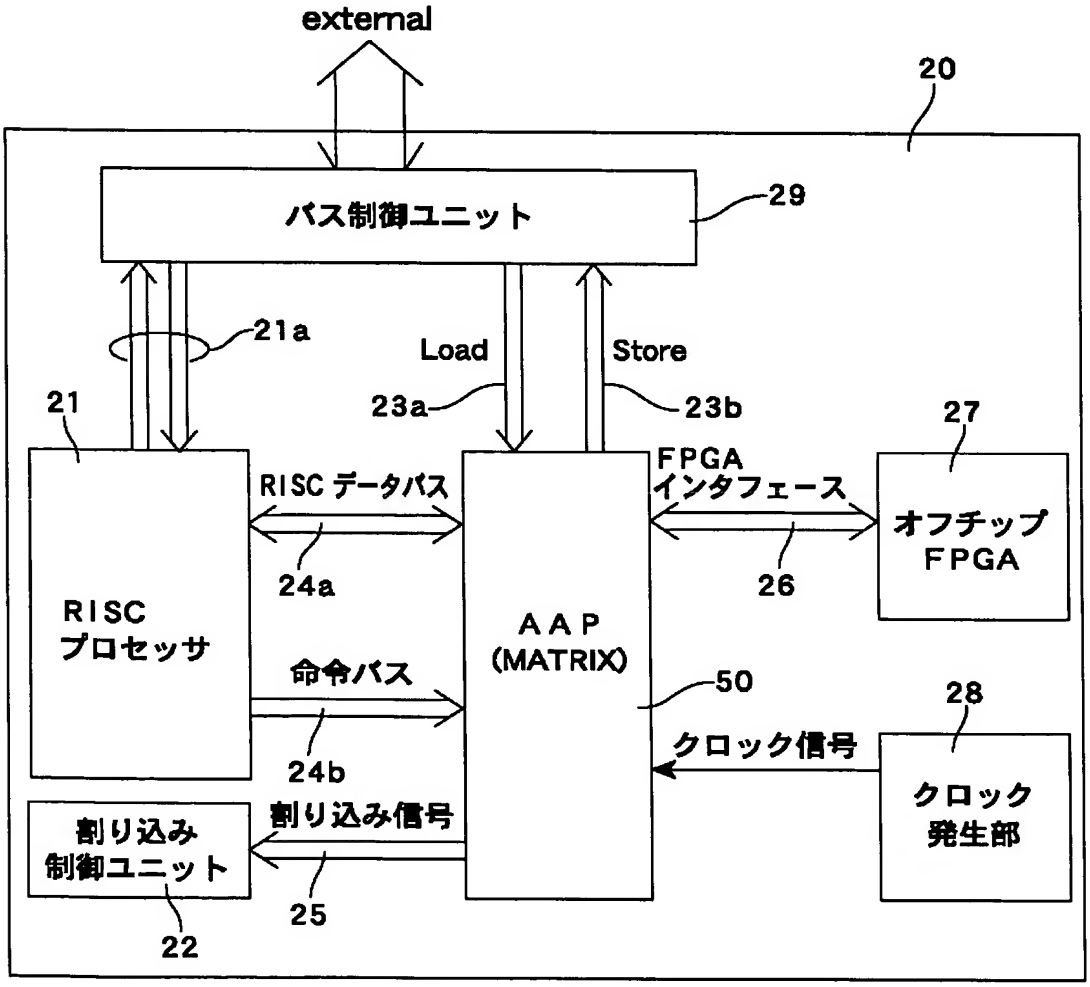
前記定義ファイルは、独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有し、前記複数の並列記述は、他の並列処理の出力データが入力されるデータ入力を少なくとも含む複数のデータ入力を備えた第1の並列処理を示す第1の並列記述を含んでおり、

前記定義ファイルに規定された前記複数の並行処理を同期して実行する工程をコンピュータにおいてシミュレートする際に、前記第1の並列処理の複数のデータ入力に、当該システムに入力されてからのレイテンシーが同一のデータを入力する、プログラム。

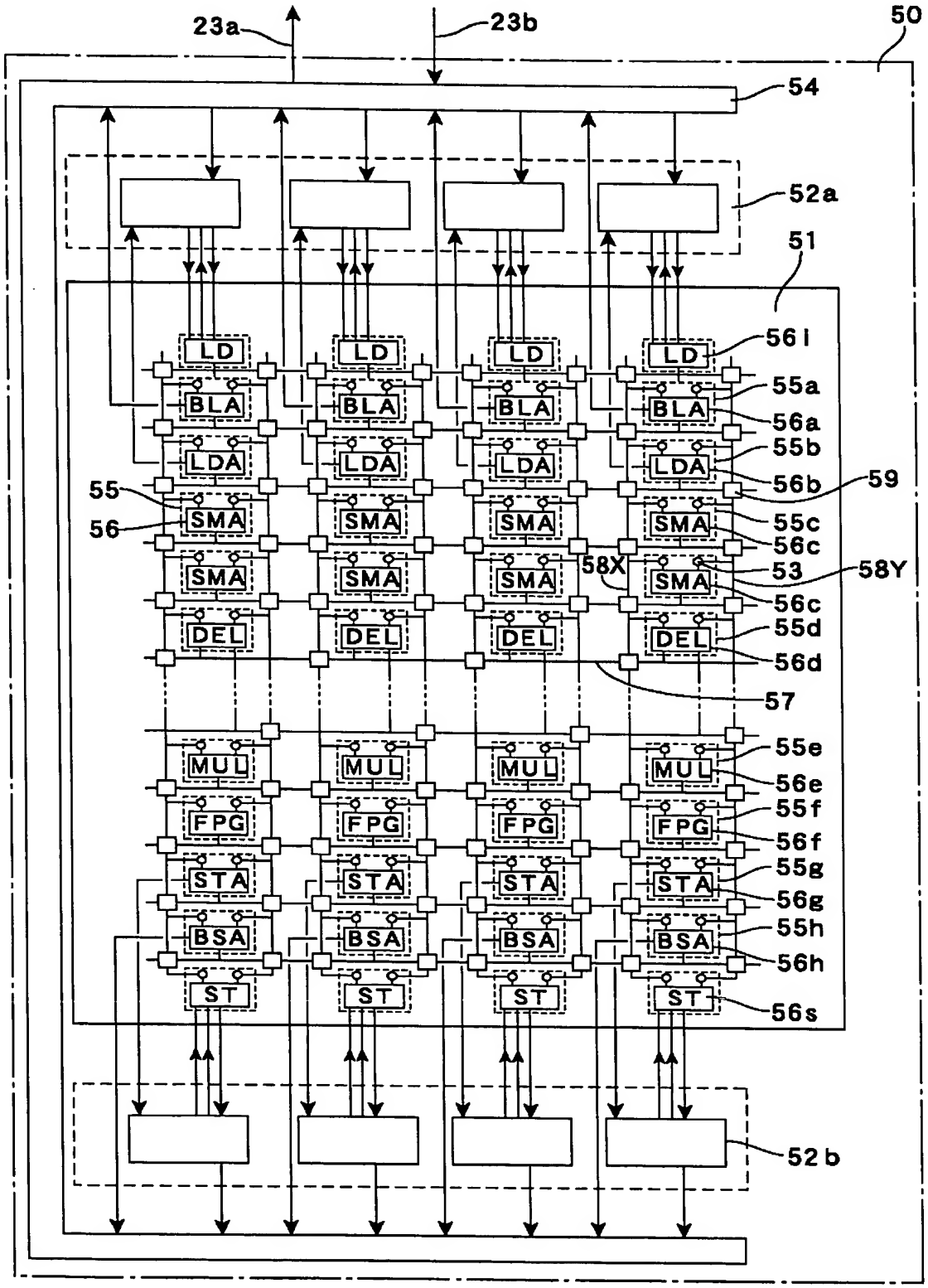
[図1]


6
3
DIDL
ハードウェア
ライブラリ
1
9
コンパイラ
5
プログラム
2
4
DDDL

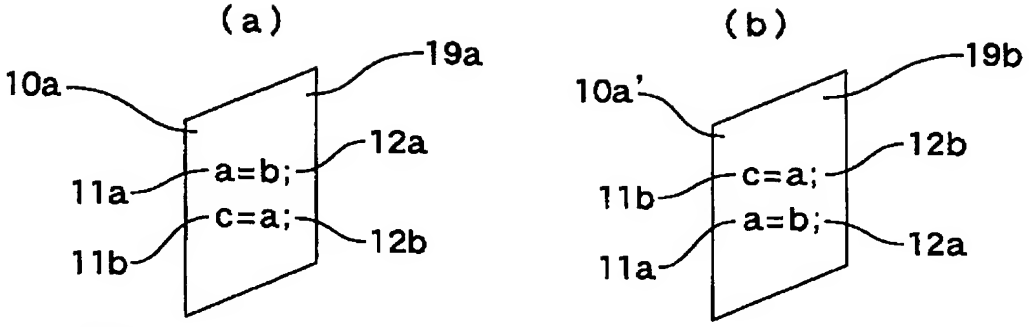
[図2]


external
20
バス制御ユニット
29
21a
Load
Store
21
23a
23b
27
RISC データバス
FPGA
インタフェース
RISC
プロセッサ
24a
AAP
(MATRIX)
26
オフチップ
FPGA
命令バス
50
28
クロック信号
24b
クロック
発生部
割り込み
制御ユニット
割り込み信号
25
22

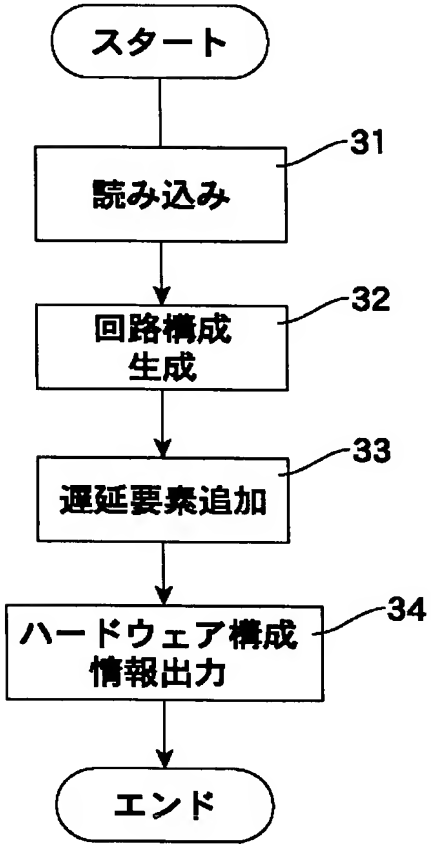
[図3]


23a
23b
50
54
52a
51
LD
BLA
LDA
SMA
DEL
MUL
FPG
STA
BSA
ST
56l
55a
56a
55b
56b
59
55
56
55c
56c
58X
53
58Y
56c
55d
56d
57
55e
56e
55f
56f
55g
56g
55h
56h
56s
52b

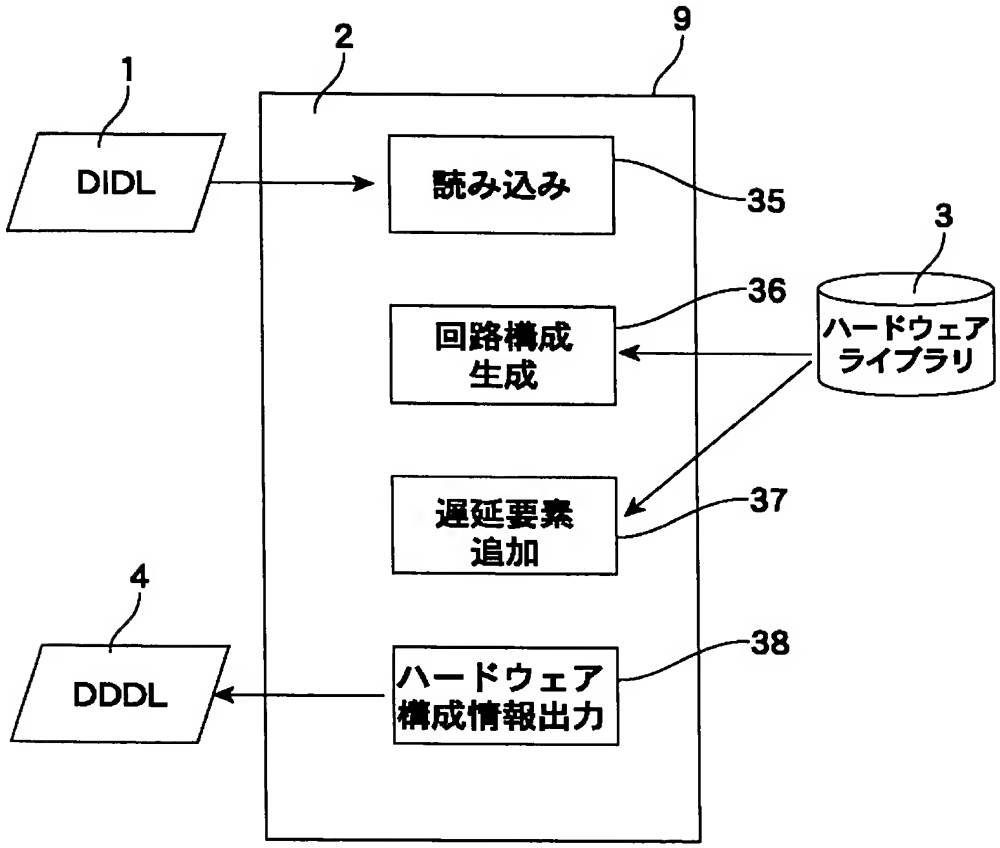
[図4]


(a)
10a
19a
a=b;
12a
11a
c=a;
11b
12b
(b)
10a'
19b
c=a;
12b
11b
a=b;
11a
12a

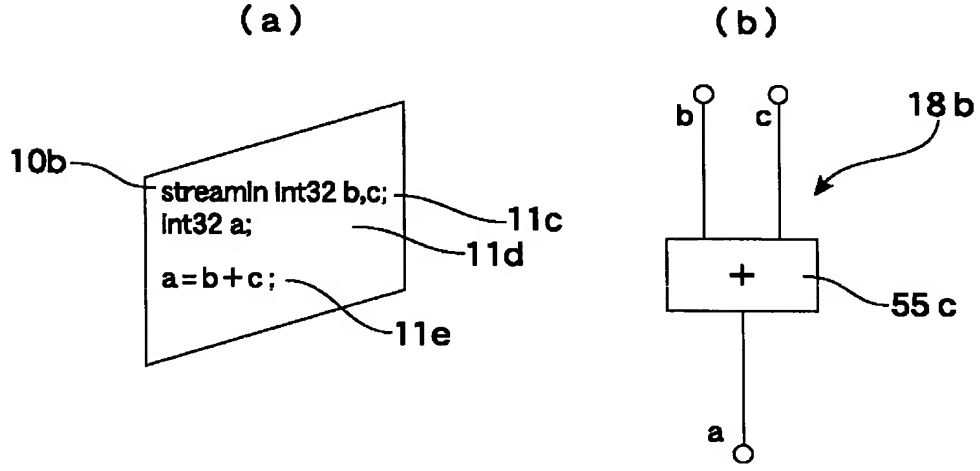
[図5]


スタート
読み込み
31
回路構成
生成
32
遅延要素追加
33
ハードウェア構成
情報出力
34
エンド

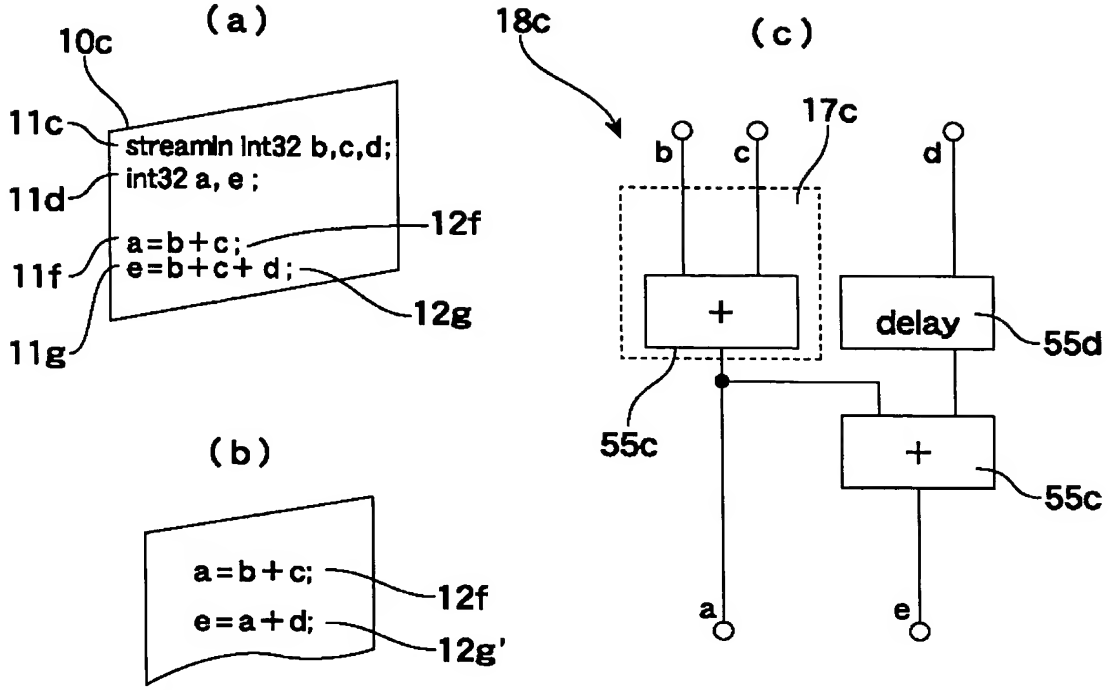
[図6]


1
2
9
DIDL
読み込み
35
3
36
回路構成
生成
ハードウェア
ライブラリ
遅延要素
追加
37
4
DDDL
ハードウェア
構成情報出力
38

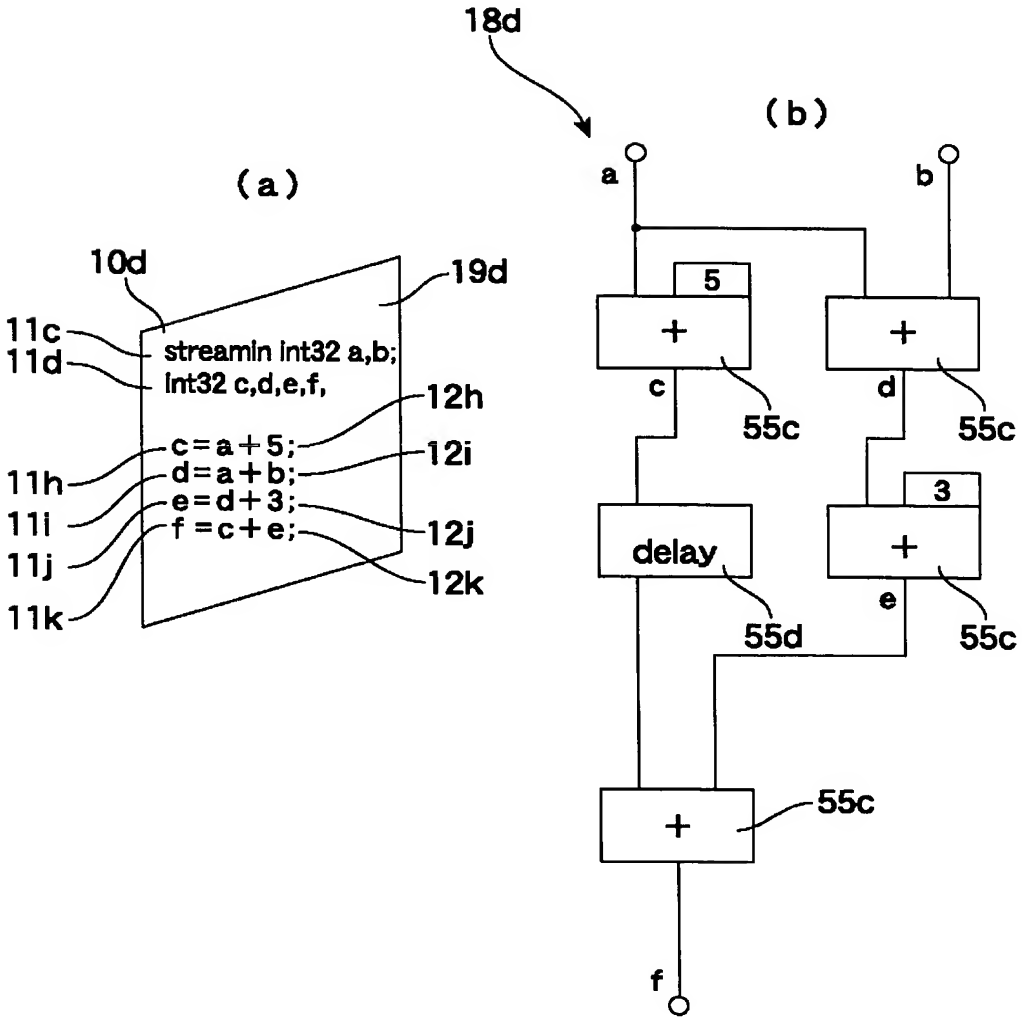
[図7]


(a)
(b)
10b
streamin int32 b,c;
int32 a;
a=b+c;
11c
11d
11e
b
c
18b
+
55c
a

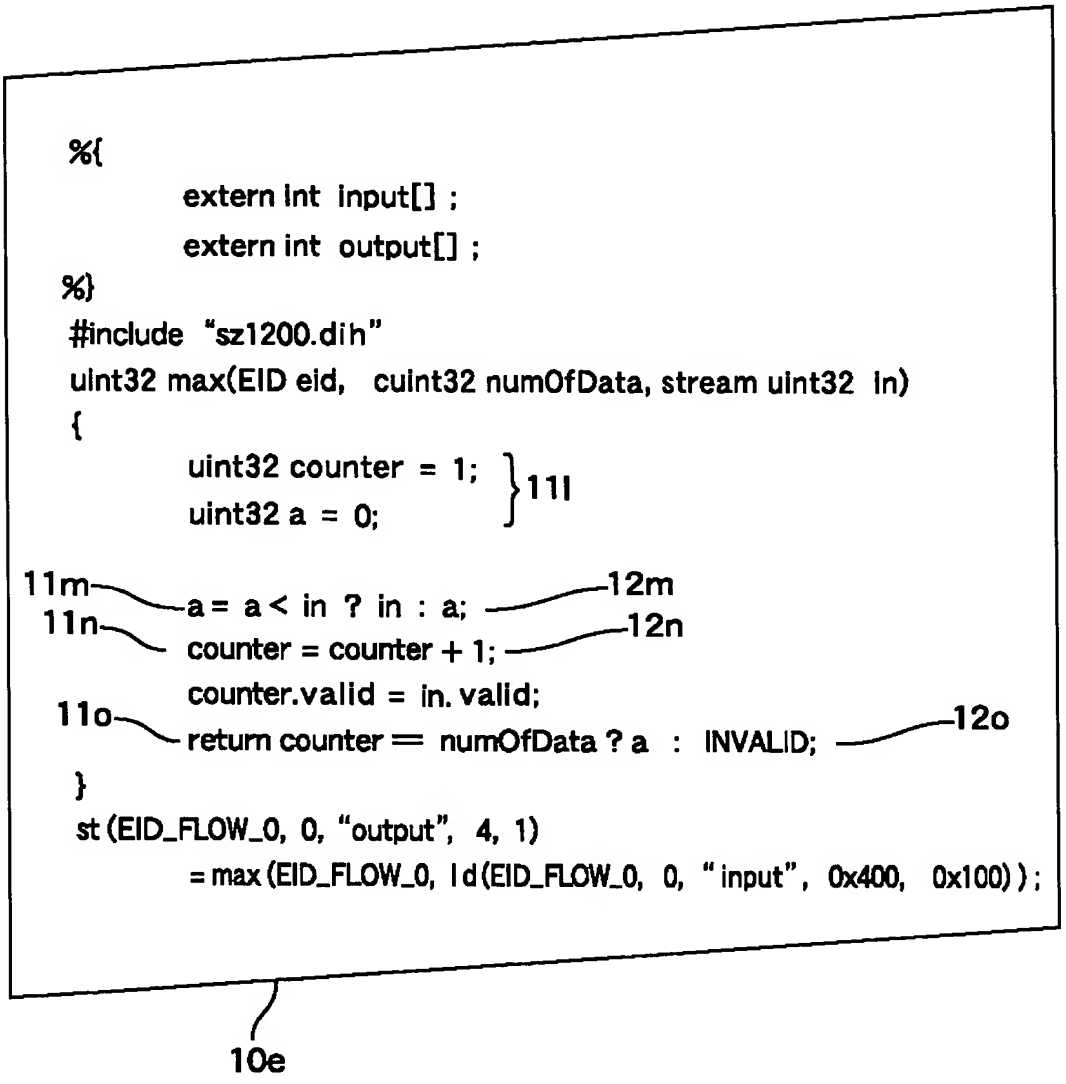
[図8]


(a)
10c
11c
streamin int32 b,c,d;
11d
int32 a, e ;
12f
11f
a=b+c;
e=b+c+ d;
12g
11g
(b)
a=b+c;
12f
e=a+d;
12g'
18c
(c)
17c
b
c
d
+
delay
55d
55c
+
55c
a
e

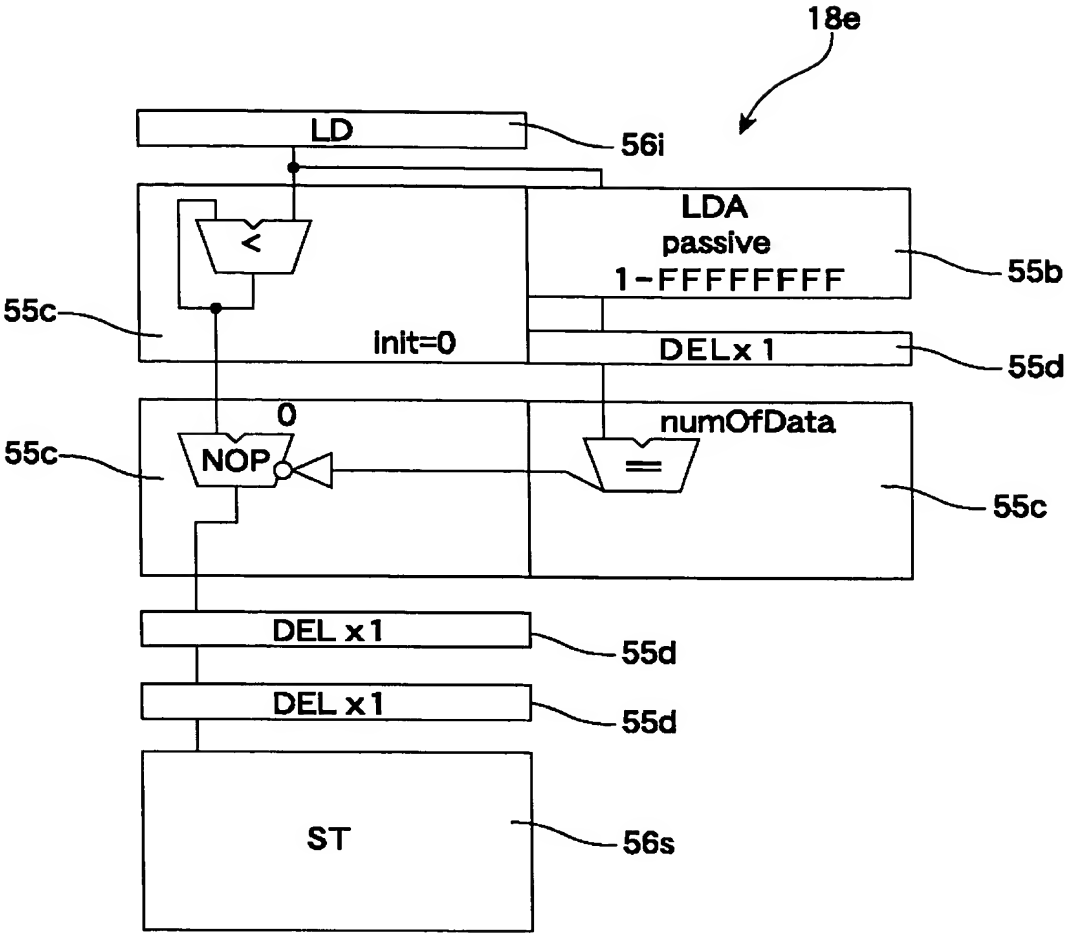
[図9]


18d
(a)
10d
19d
11c
11d
streamin int32 a,b;
int32 c,d,e,f,
12h
c=a+5;
12i
11h
d=a+b;
11i
e=d+3;
12j
11j
f =c+e;
12k
11k
(b)
a
b
5
+
+
c
55c
d
55c
3
delay
+
e
55d
55c
+
55c
f

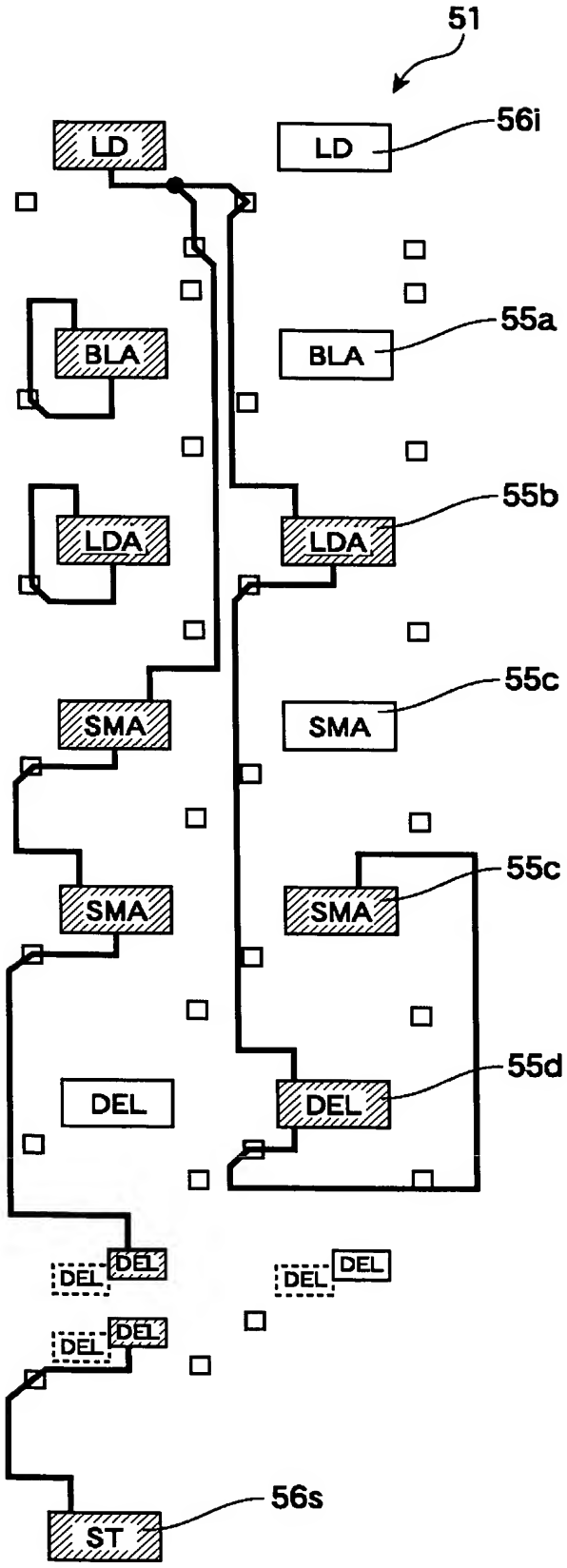
[図10]


%{
extern int input[] ;
extern int output[] ;
%}
#include "sz1200.dih"
uint32 max(EID eid, cuint32 numOfData, stream uint32 in)
{
uint32 counter = 1;
uint32 a = 0;
11l
11m
a = a < in ? in : a;
12m
11n
counter = counter + 1;
12n
counter.valid = in.valid;
11o
return counter == numOfData ? a : INVALID;
12o
}
st (EID_FLOW_0, 0, "output", 4, 1)
= max (EID_FLOW_0, Id(EID_FLOW_0, 0, "input", 0x400, 0x100));
10e

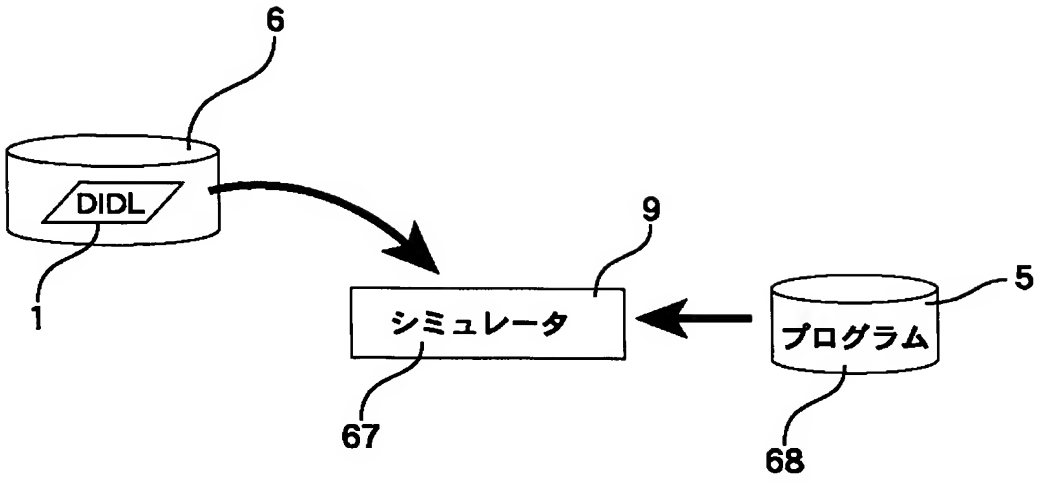
[図11]


18e
LD
56i
LDA
passive
1-FFFFFFFF
55b
<
55c
init=0
DELx 1
55d
0
numOfData
55c
NOP
=
55c
DEL x1
55d
DEL x1
55d
ST
56s

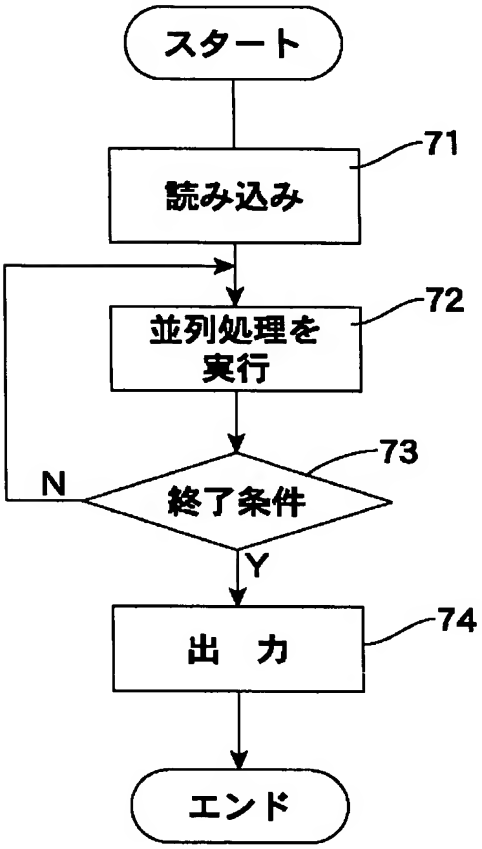
[図12]


51
LD
LD
56i
BLA
BLA
55a
LDA
LDA
55b
SMA
SMA
55c
SMA
SMA
55c
DEL
DEL
55d
DEL
DEL
DEL
DEL
DEL
ST
56s

[図13]


6
DIDL
1
9
シミュレータ
67
5
プログラム
68

[図14]


スタート
71
読み込み
72
並列処理を
実行
73
N
終了条件
Y
74
出　力
エンド

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2004/009000

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl[7] G06F17/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl[7] G06F17/50

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Toshinori SUEYOSHI, "Reconfigurable Computing System no Genjo to Kadai -Computer Evolution e Mukete-", The Institute of Electronics, Information and Communication Engineers Gijutsu Kenkyu Hokoku, The Institute of Electronics, Information and Communication Engineers, JP, Vol.96, No.426, pages 111 to 118 (CPSY96-91), 13 December, 1996 (13.12.96), page 117, right column, lines 22 to 55	1-14

☐ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
09 August, 2004 (09.08.04)	31 August, 2004 (31.08.04)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

Form PCT/ISA/210 (second sheet) (January 2004)

国際調査報告

国際出願番号　PCT／JP2004／009090

A.　発明の属する分野の分類（国際特許分類（IPC））

Int. Cl⁷ G06F17／50

B.　調査を行った分野

調査を行った最小限資料（国際特許分類（IPC））

Int. Cl⁷ G06F17／50

最小限資料以外の資料で調査を行った分野に含まれるもの

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C.　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
A	末吉敏則、Reconfigurable Computing System の現状と課題 -Computer Evolution へ向けて-、電子情報通信学会技術研究報告、社団法人電子情報通信学会、日本、vol.96, no.426, p111-118 (CPSY96-91)、1996.12.13, p117右欄22-55行	1－14

☐ C欄の続きにも文献が列挙されている。　☐ パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー
「A」特に関連のある文献ではなく、一般的技術水準を示すもの
「E」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「O」口頭による開示、使用、展示等に言及する文献
「P」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「&」同一パテントファミリー文献

国際調査を完了した日	国際調査報告の発送日
09. 08. 2004	31. 8. 2004

国際調査機関の名称及びあて先	特許庁審査官（権限のある職員）	
日本国特許庁（ISA／JP） 郵便番号100－8915 東京都千代田区霞が関三丁目4番3号	田中　幸雄	5H　9191
	電話番号　03－3581－1101　内線　3531	

様式PCT／ISA／210（第2ページ）（2004年1月）